TOSCA

Topology and Orchestration Specification for Cloud Applications (TOSCA) Standard

OASIS TOSCA presentation to ETSI NFV Information Modelling Workshop

Proposal for way forward-How TOSCA Adds Value in NFV world

Agenda

- **TOSCA Overview**
- **Way forward-How TOSCA can be used in NFV**
  - Many problems can be solved with TOSCA
- **Some Open Source Implementations**
  - Openstack, Tacker, Parser, Senlin, alien4cloud, Cloudify
- **Main Features**
  - containers, policies， network modeling,
- **Backup slides**
  - Sample template
  - Lifecycle sequency

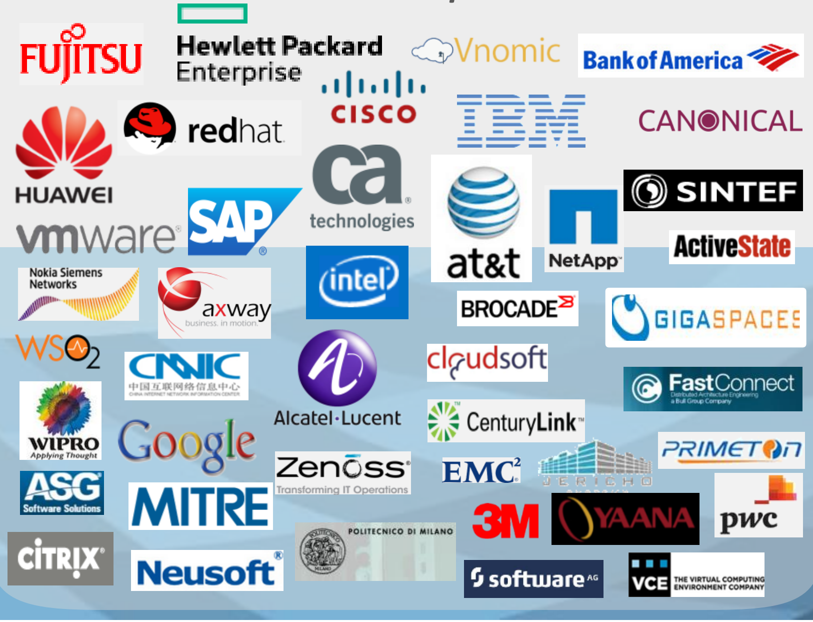
TOSCA Overview

What is TOSCA?

- **TOSCA** is an important **new open cloud standard,** that is enabling a unique eco-system, supported by a large and growing number of international industry leaders…

- **TOSCA Version 1.0 Specification approved as an OASIS Standard** *(Nov 2013)*
- **Government and Corporate Awareness**:
  - **OASIS**: 600+ participant organizations.
    5000+ participants spanning 65+ countries
  - **TOSCA Committee**: 170+ people 45+ companies/orgs
  - **International Standards & Research**: ETSI NFV liaison, EU FP7, etc.
  - **Industry Analysts**: Forrester names TOSCA as a top four cloud open standard *(Mar 2014)*
- **Multi-company Interoperability Demonstrated**:
  - **EuroCloud 2013** *(Oct 2013)*: IBM, SAP, Fujitsu, Huawei, HP, Vnomic, Zenoss and others
  - **Open Data Center Alliance**: [TOSCA Application Portability in the Enterprise Cloud](#) PoC *(Jan 2014)*

Associated Companies



Way forward-How TOSCA can be used in NFV

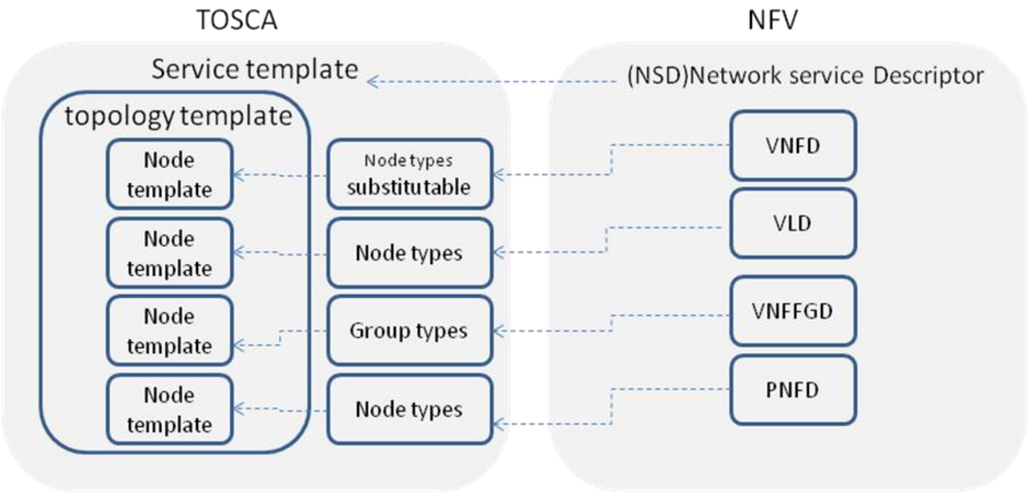
Many problems can be solved with TOSCA

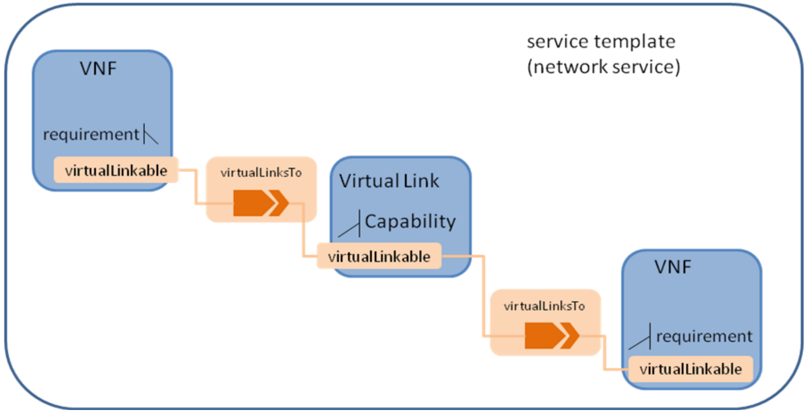
- ***Topology***
- ***Composition***
- ***Lifecycle***
- ***Portability***

Topology

As the Topology and Orchestration Specification for Cloud Applications, TOSCA is mainly used to describe the topology of the deployment view for cloud applications.

- *Defining node templates to describe components in the topology structure*
- *Defining relationship templates to describe connecting, dependency, deployment ordering*



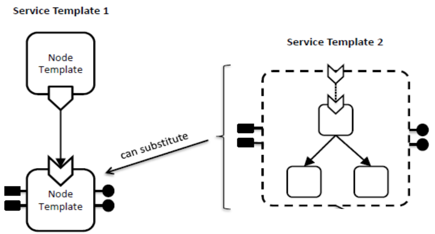


TOSCA can be used to describe the topology of a Network service or VNF as defined by ETSI NFV.

VNF, VL can be defined as node templates in TOSCA. A new virtualLinksTo relationship type can be defined to connect VNF and VL.

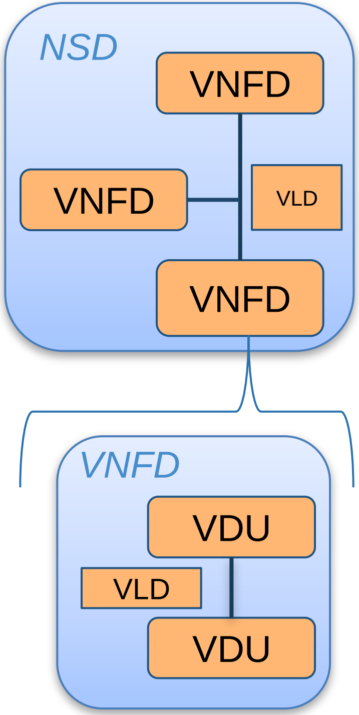
Composition

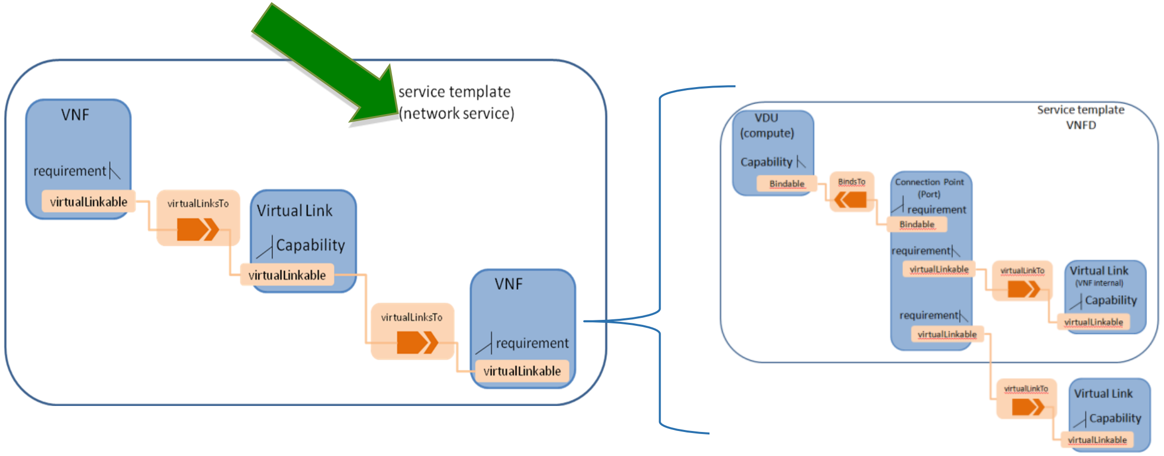
Any node in a TOSCA topology can be an abstraction of another layer or subtopology



NFV information model has the layered structure.

- ***NSD are composed of VNFD, VLD, PNFD and etc.***
- ***VNFD are composed of VDU, VLD and etc.***





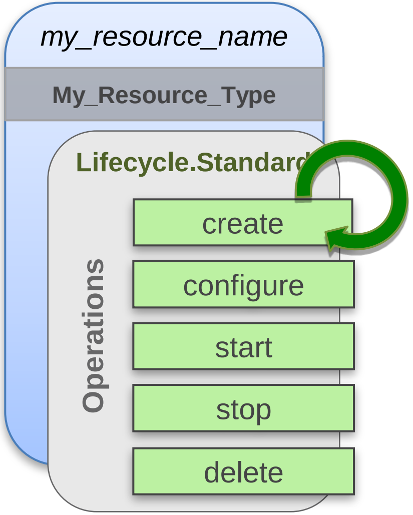
Using the TOSCA substitution feature, NFV information model can be described by using multiple TOSCA service templates

Lifecycle

TOSCA models have a consistent view of state-based lifecycle

- ✓ have **Operations** (implementations) that can be sequenced against state of any dependent resources
- ✓ *fits into any **Management Framework** or **Access Control System***

Standardize Resource Lifecycle



my_resource_name

My_Resource_Type

Lifecycle.Standard

Operations

create

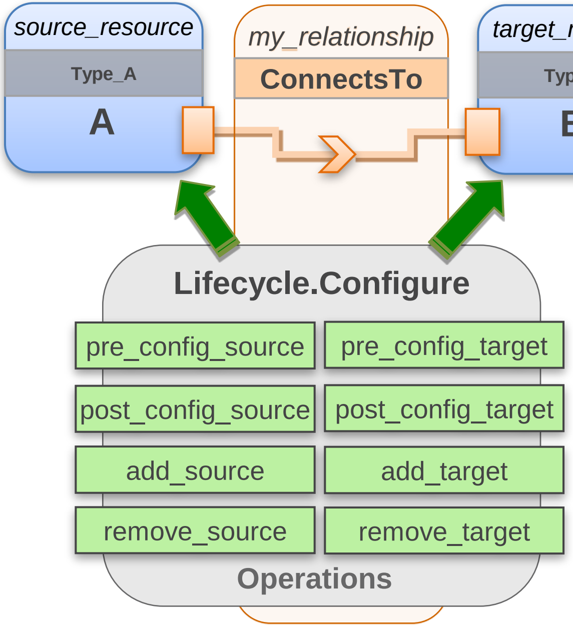
configure

start

stop

delete

Standardize Relationship Lifecycle



source_resource

Type_A

A

my_relationship

ConnectsTo

target_resource

Type_B

B

Lifecycle.Configure

pre_config_source

pre_config_target

post_config_source

post_config_target

add_source

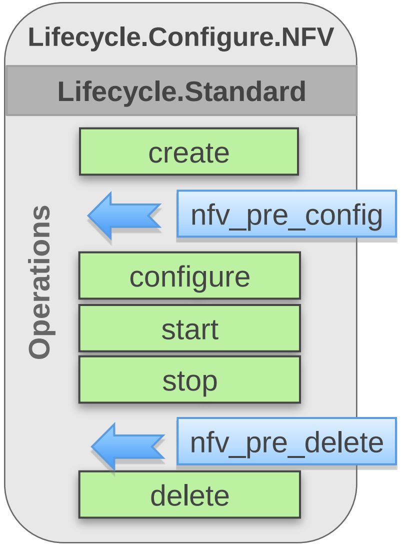
add_target

remove_source

remove_target

Operations

Lifecycle Customization



Lifecycle.Configure.NFV

Lifecycle.Standard

Operations

create

nfv_pre_config

configure

start

stop

nfv_pre_delete

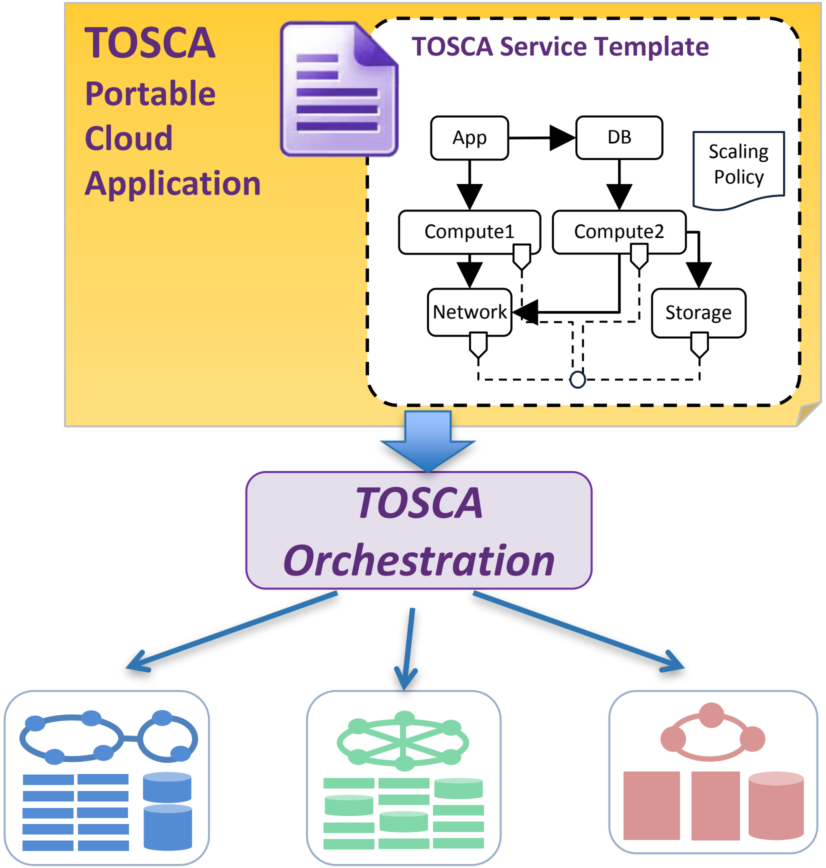
delete

Create new NFV Lifecycles or Augment existing (via subclassing)

- ✓ **Parameters** and **Policies** can be supplied to operations to affect resource behavior (state)
- ✓ **Workflow** - TOSCA is developing workflow to allow handling complex state changes, configurations, etc.

TOSCA Lifecycle can be customized for NFV Resources and Relationships

Portability



TOSCA Portable Cloud Application

TOSCA Service Template

TOSCA Orchestration

By expressing application requirements independently from cloud capabilities and implementation,
TOSCA provides:

- *Multi VIM Support*
- *Portability of services across clouds*
- *Declarative model spanning infrastructure and service*
- *Manipulate the orchestration declaratively instead of dealing with disparate cloud APIs (leave that to the TOSCA Orchestrator)*

TOSCA enables NFV applications flexible movement between different cloud infrastructures.

Some Open Source Implementations

- Senlin
- Tacker
- Parser
- Alien4cloud
- Cloudify

TOSCA in OpenSource

Openstack

Heat-translator

Deployment TemplateTranslation

Senlin

Policy (Placement&scaling)

Tacker

Opensource MANO

Cloudify

Service Orchestration & Management

OPNFV
Parser

Deployment TemplateTranslation

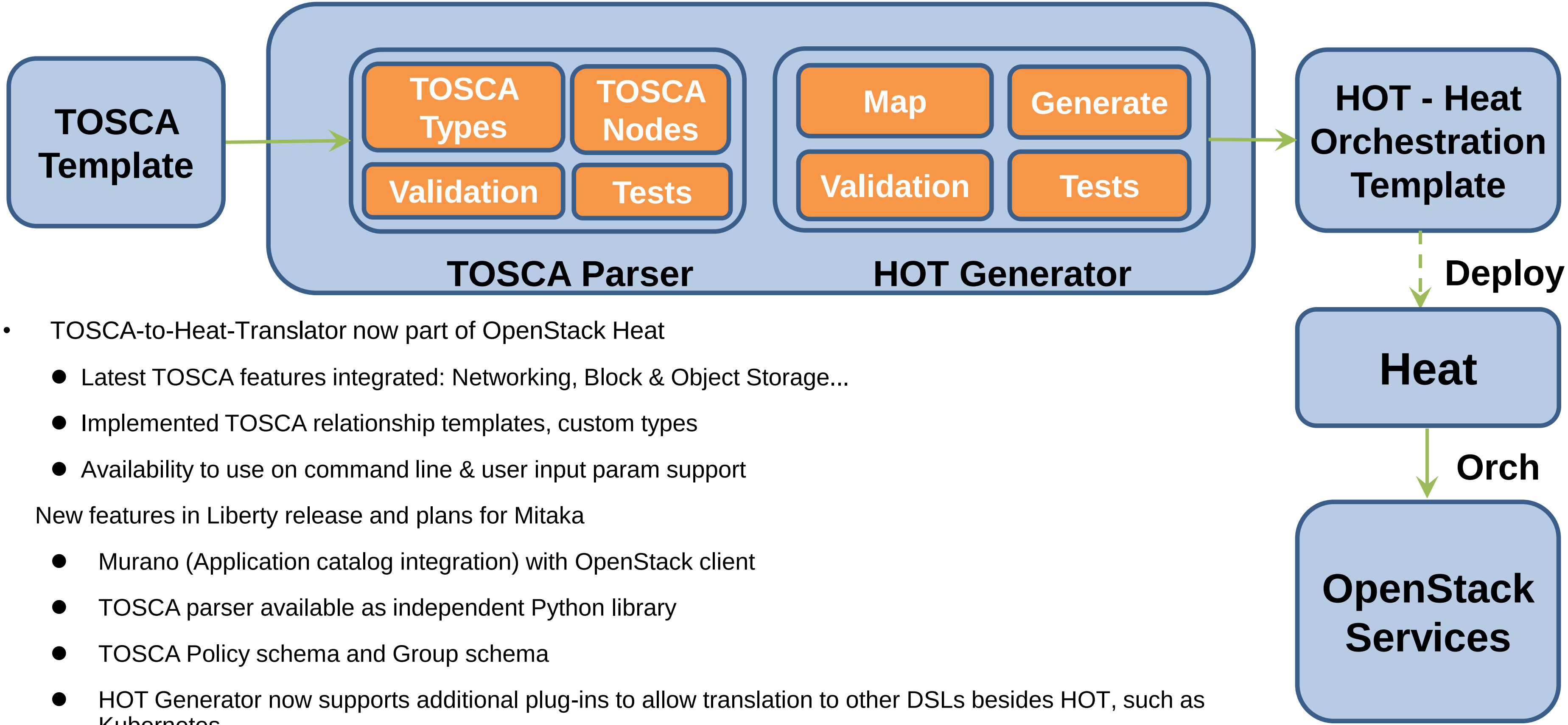
seaclouds

www.seaclouds-project.eu/media.html

alien4cloud

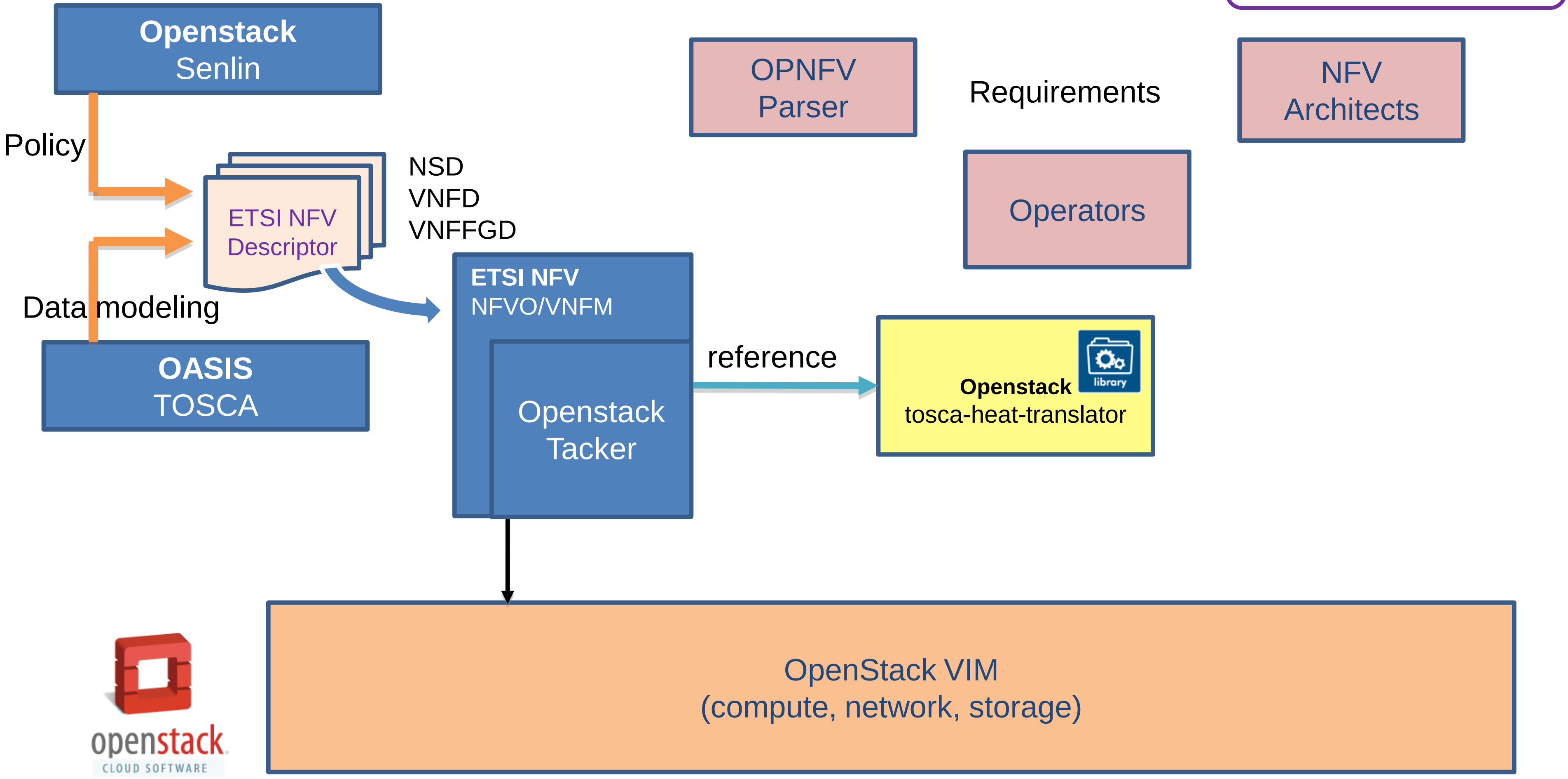
Topology & LCM Design
http://alien4cloud.github.io/

Automated TOSCA-based Orchestration Now Part of OpenStack



- TOSCA-to-Heat-Translator now part of OpenStack Heat
  - Latest TOSCA features integrated: Networking, Block & Object Storage...
  - Implemented TOSCA relationship templates, custom types
  - Availability to use on command line & user input param support
- New features in Liberty release and plans for Mitaka
  - Murano (Application catalog integration) with OpenStack client
  - TOSCA parser available as independent Python library
  - TOSCA Policy schema and Group schema
  - HOT Generator now supports additional plug-ins to allow translation to other DSLs besides HOT, such as Kubernetes

OpenSource related to ETSI NFV and OASIS TOSCA



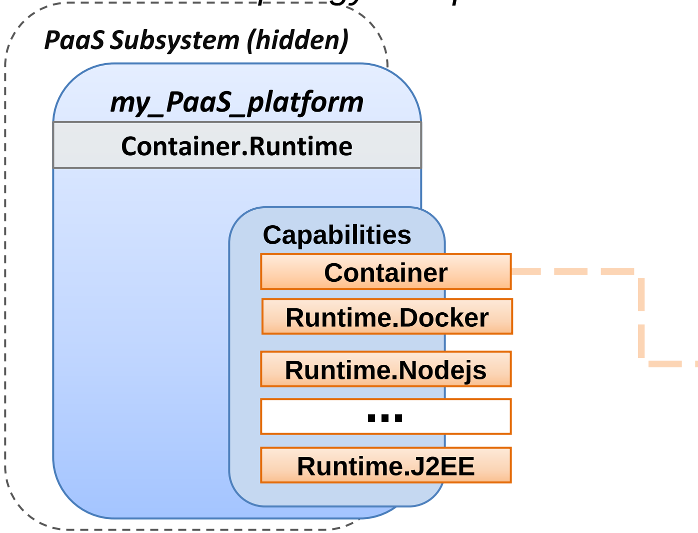
Main features

- Container
- Policies
- Network modeling

TOSCA Model for Containers leveraging Repositories

PaaS Modeling

- *Template author chooses to expose or hide runtime topology & implementation*

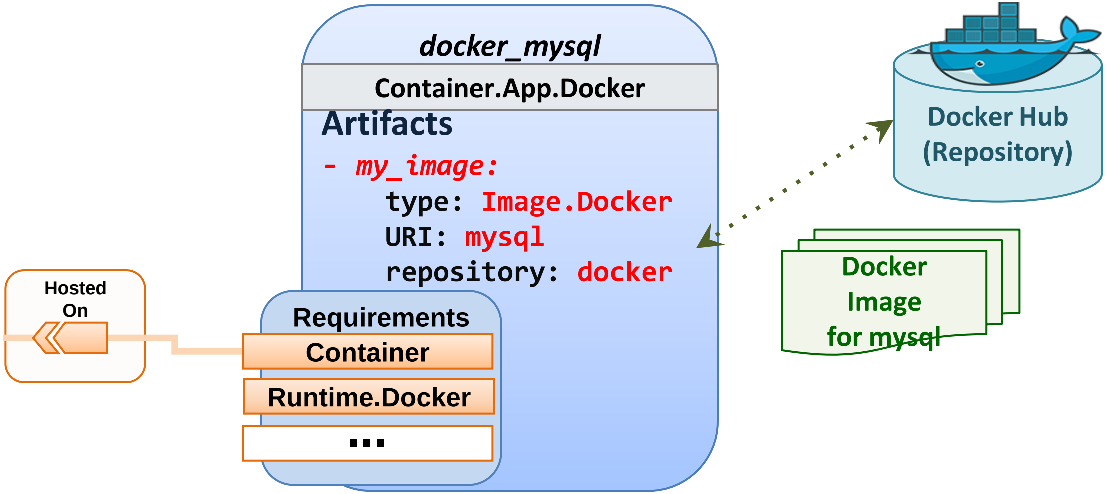


PaaS Layer exposes "runtimes" as TOSCA Capabilities

- *Docker, Nodejs, JSP, J2EE, etc.*

Container Application Modeling

- ***Agnostic of PaaS Cloud Provider***
  - *PaaS on OpenStack, Cloud Foundry, Azure, etc.*



```
- my_image:
    type: Image.Docker
    URI: mysql
    repository: docker
```

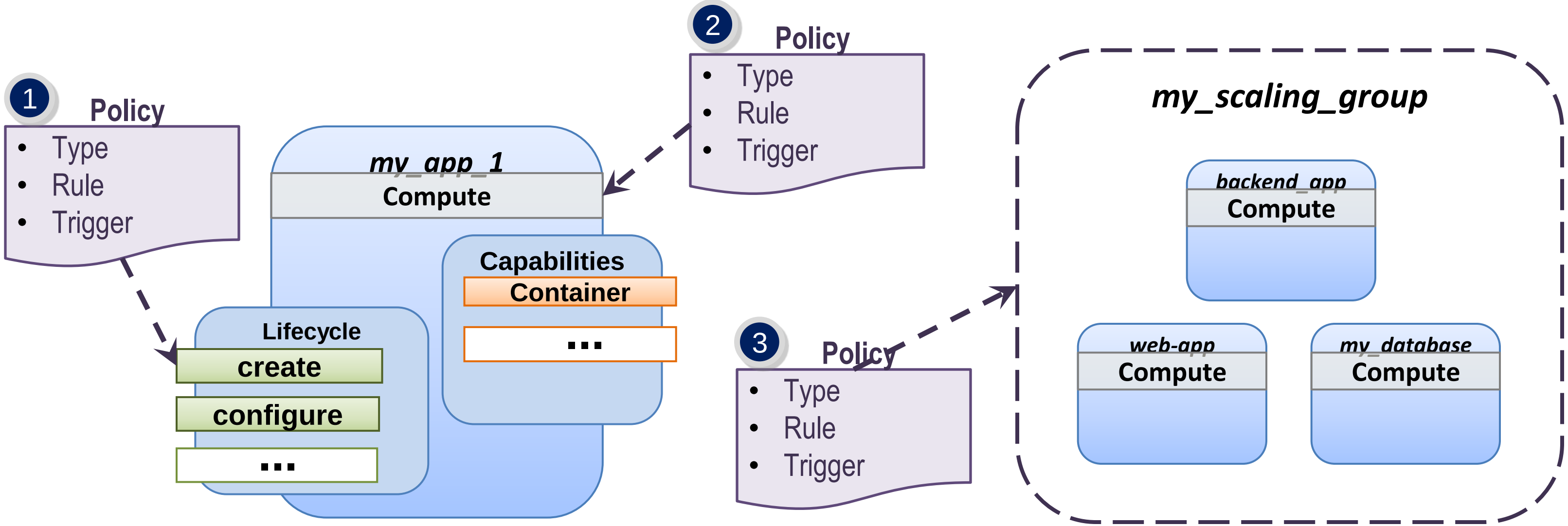
Orchestrators could automatically retrieve and deploy a Docker image from a declared Repository

- *TOSCA Templates can model repositories*
- *Orchestrators could dynamically "pull" from multiple repositories*

TOSCA Direction to model Policies

Supported areas: **Placement** (Affinity), **Scaling** and **Performance**

- *with **Rules** that are evaluated to execute Automatic and Imperative **Triggers***



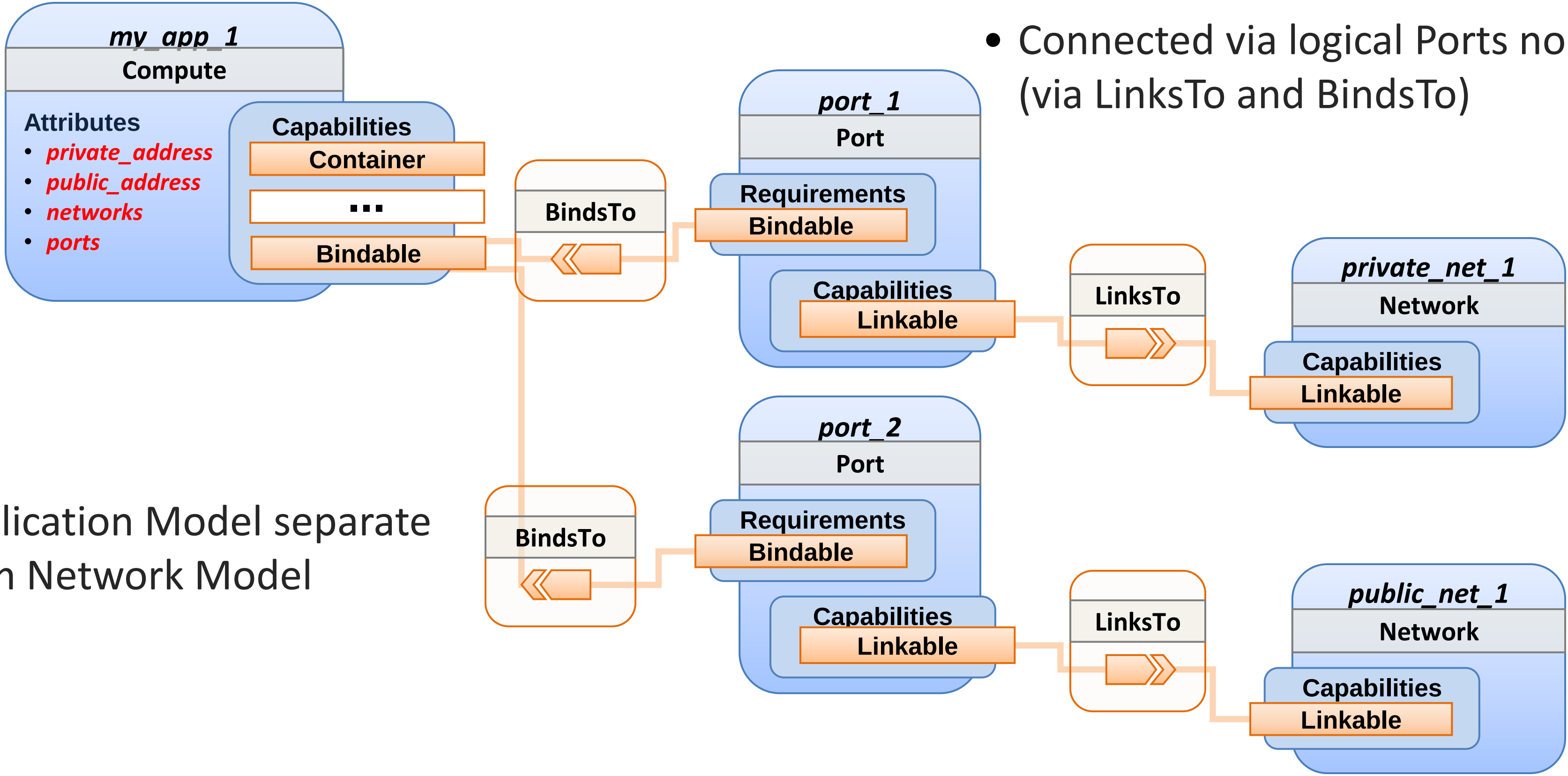
Policies modeled as *Requirements* using *Capability Types that can be attached to*

- *Interfaces or specific Operations*
- *Nodes and*
- *Groups of Nodes*

TOSCA defines policies using an Event-Condition-Action model

TOSCA Model for Logical Public & Private Cloud Networks

- Connected via logical Ports nodes (via LinksTo and BindsTo)



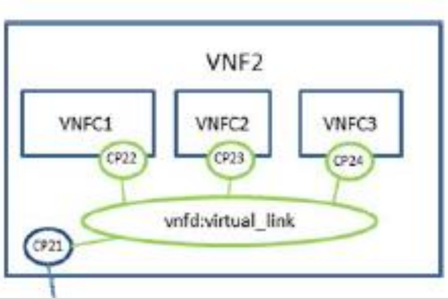
- Application Model separate from Network Model

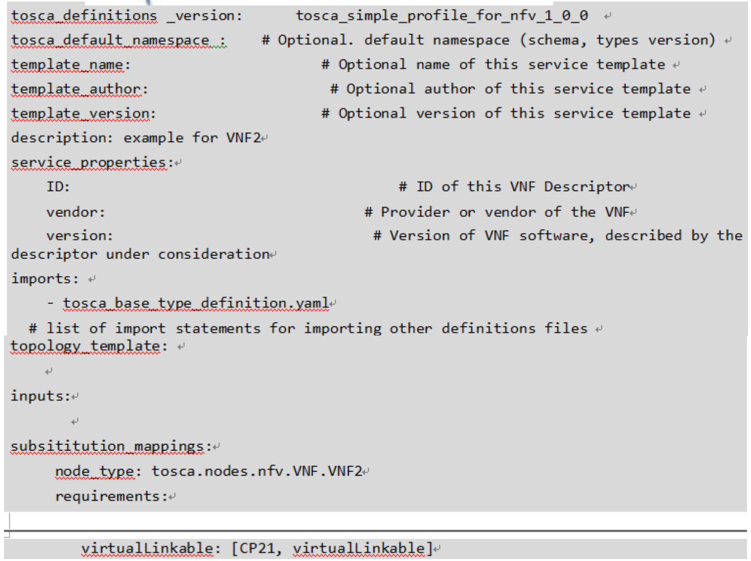
Allows developers to model JUST the application bind to existing tenant networks

Backup slides

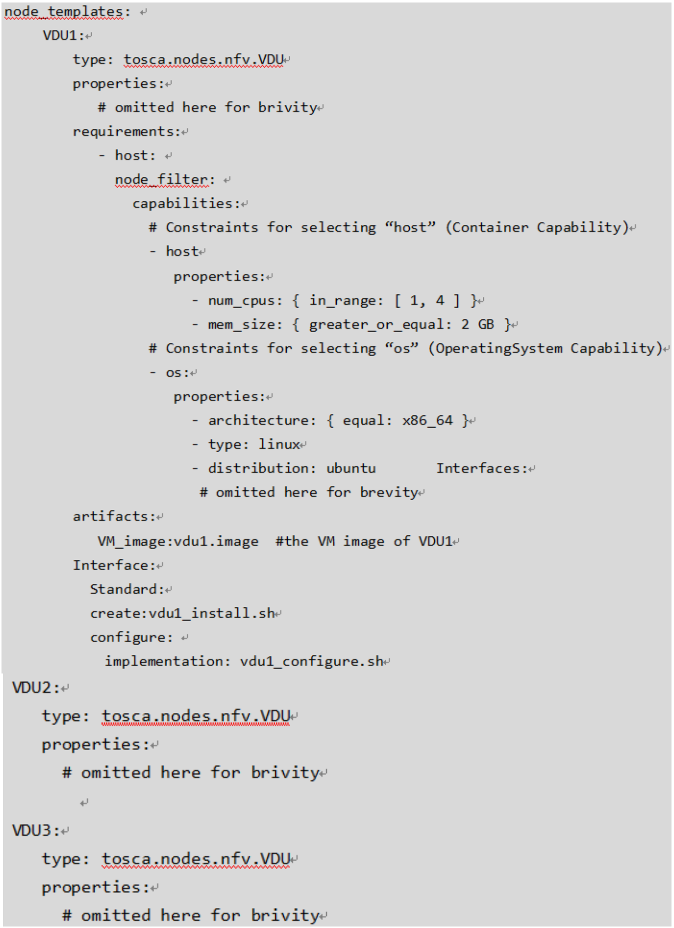
- sample template
- Lifecycle Sequencing

VNFD example

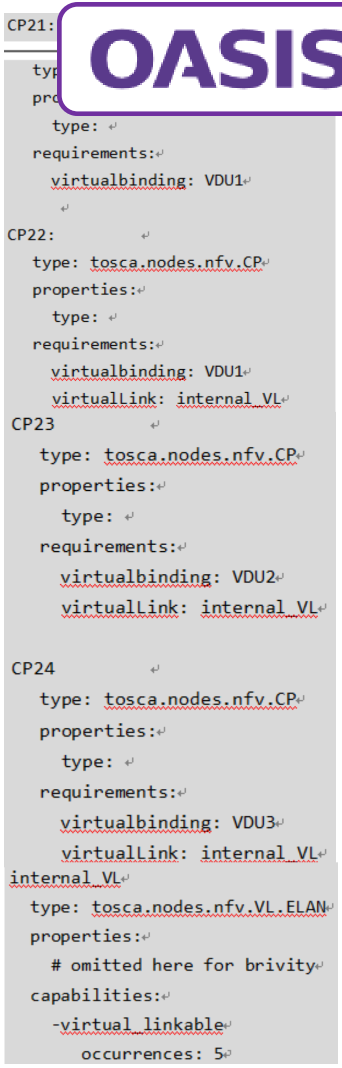




```
tosca_definitions _version:     tosca_simple_profile_for_nfv_1_0_0
tosca_default_namespace :    # Optional. default namespace (schema, types version)
template_name:                     # Optional name of this service template
template_author:                    # Optional author of this service template
template_version:                  # Optional version of this service template
description: example for VNF2
service_properties:
    ID:                                       # ID of this VNF Descriptor
    vendor:                               # Provider or vendor of the VNF
    version:                               # Version of VNF software, described by the
descriptor under consideration
imports:
    - tosca_base_type_definition.yaml
  # list of import statements for importing other definitions files
topology_template:

inputs:

subsititution_mappings:
     node_type: tosca.nodes.nfv.VNF.VNF2
     requirements:

        virtualLinkable: [CP21, virtualLinkable]
```

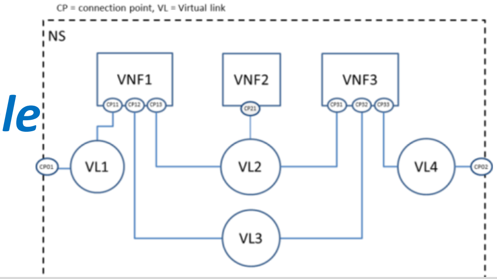


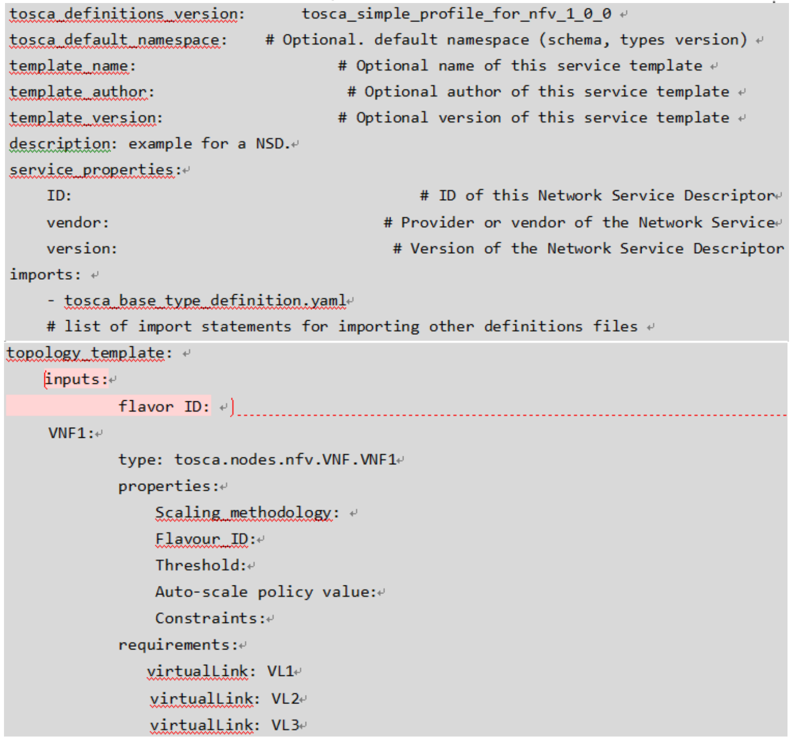
```
node_templates:
    VDU1:
       type: tosca.nodes.nfv.VDU
       properties:
          # omitted here for brivity
       requirements:
          - host:
            node_filter:
              capabilities:
                # Constraints for selecting “host” (Container Capability)
                - host
                   properties:
                     - num_cpus: { in_range: [ 1, 4 ] }
                     - mem_size: { greater_or_equal: 2 GB }
                # Constraints for selecting “os” (OperatingSystem Capability)
                - os:
                   properties:
                     - architecture: { equal: x86_64 }
                     - type: linux
                     - distribution: ubuntu       Interfaces:
                      # omitted here for brevity
       artifacts:
          VM_image:vdu1.image  #the VM image of VDU1
       Interface:
         Standard:
         create:vdu1_install.sh
         configure:
           implementation: vdu1_configure.sh
 VDU2:
    type: tosca.nodes.nfv.VDU
    properties:
      # omitted here for brivity

 VDU3:
    type: tosca.nodes.nfv.VDU
    properties:
      # omitted here for brivity
```



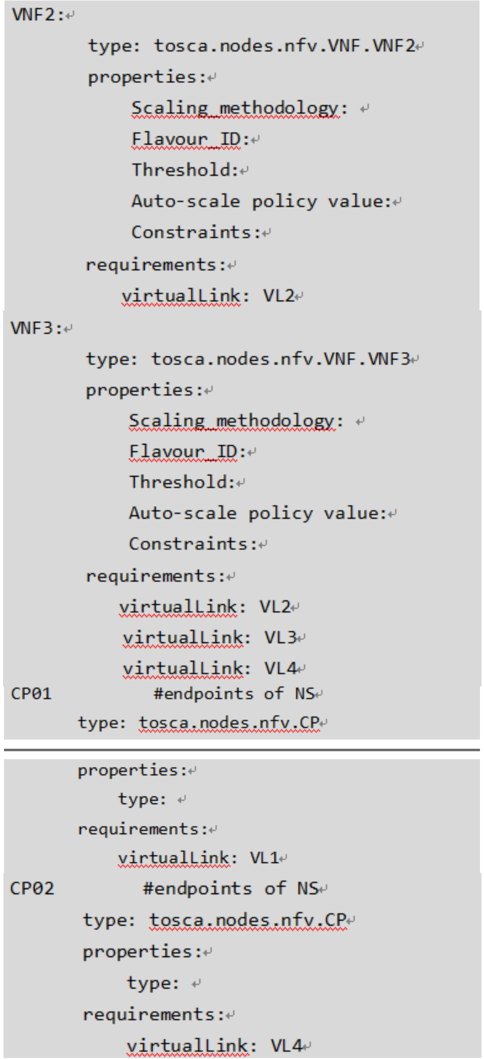
```
CP21:
   typ
   pro
     type:
   requirements:
     virtualbinding: VDU1

CP22:
   type: tosca.nodes.nfv.CP
   properties:
     type:
   requirements:
     virtualbinding: VDU1
     virtualLink: internal_VL
 CP23
    type: tosca.nodes.nfv.CP
    properties:
      type:
    requirements:
      virtualbinding: VDU2
      virtualLink: internal_VL

 CP24
    type: tosca.nodes.nfv.CP
    properties:
      type:
    requirements:
      virtualbinding: VDU3
      virtualLink: internal_VL
internal_VL
   type: tosca.nodes.nfv.VL.ELAN
   properties:
     # omitted here for brivity
   capabilities:
     -virtual_linkable
        occurrences: 5
```

NSD example

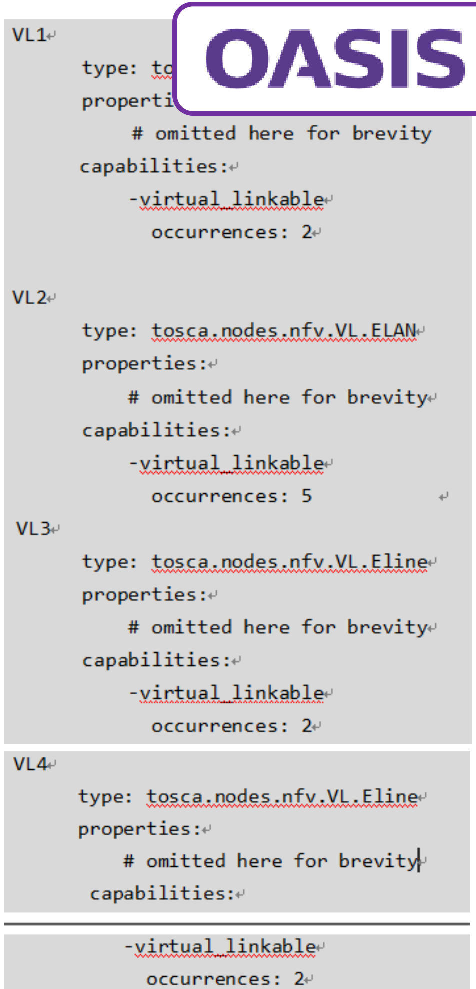




```
tosca_definitions_version:      tosca_simple_profile_for_nfv_1_0_0
tosca_default_namespace:    # Optional. default namespace (schema, types version)
template_name:                      # Optional name of this service template
template_author:                     # Optional author of this service template
template_version:                   # Optional version of this service template
description: example for a NSD.
service_properties:
    ID:                                          # ID of this Network Service Descriptor
    vendor:                                  # Provider or vendor of the Network Service
    version:                                  # Version of the Network Service Descriptor
imports:
    - tosca_base_type_definition.yaml
    # list of import statements for importing other definitions files
topology_template:
    inputs:
            flavor ID:
    VNF1:
            type: tosca.nodes.nfv.VNF.VNF1
            properties:
                Scaling methodology:
                Flavour ID:
                Threshold:
                Auto-scale policy value:
                Constraints:
            requirements:
               virtualLink: VL1
               virtualLink: VL2
               virtualLink: VL3
```



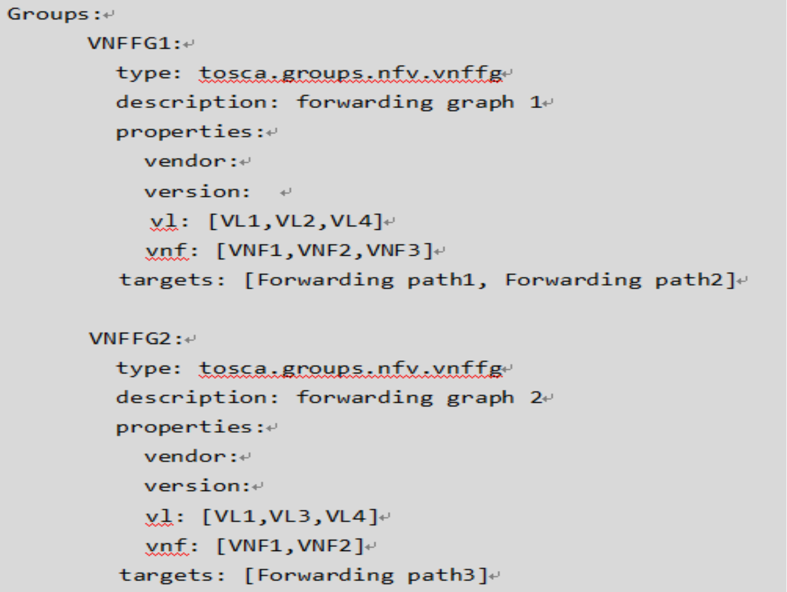
```
VNF2:
        type: tosca.nodes.nfv.VNF.VNF2
        properties:
            Scaling methodology:
            Flavour ID:
            Threshold:
            Auto-scale policy value:
            Constraints:
        requirements:
            virtualLink: VL2
VNF3:
        type: tosca.nodes.nfv.VNF.VNF3
        properties:
            Scaling methodology:
            Flavour ID:
            Threshold:
            Auto-scale policy value:
            Constraints:
        requirements:
            virtualLink: VL2
            virtualLink: VL3
            virtualLink: VL4
CP01            #endpoints of NS
       type: tosca.nodes.nfv.CP
       properties:
           type:
       requirements:
           virtualLink: VL1
CP02            #endpoints of NS
       type: tosca.nodes.nfv.CP
       properties:
           type:
       requirements:
           virtualLink: VL4
```



```
VL1
       type: to
       properti
            # omitted here for brevity
       capabilities:
            -virtual_linkable
              occurrences: 2

VL2
       type: tosca.nodes.nfv.VL.ELAN
       properties:
            # omitted here for brevity
       capabilities:
            -virtual_linkable
              occurrences: 5
VL3
       type: tosca.nodes.nfv.VL.Eline
       properties:
            # omitted here for brevity
       capabilities:
            -virtual_linkable
              occurrences: 2
VL4
       type: tosca.nodes.nfv.VL.Eline
       properties:
            # omitted here for brevity
        capabilities:
            -virtual_linkable
              occurrences: 2
```

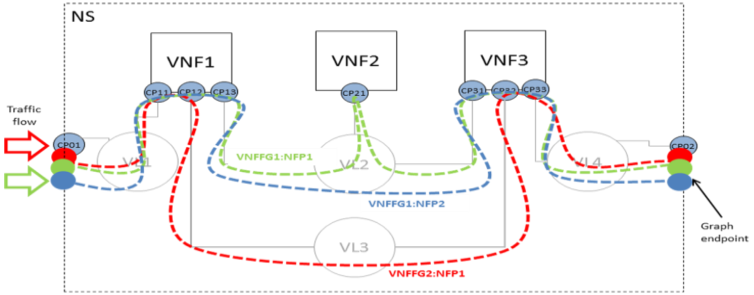
VNFFG

–Using TOSCA group concept



```
Groups:
      VNFFG1:
        type: tosca.groups.nfv.vnffg
        description: forwarding graph 1
        properties:
          vendor:
          version:
           vl: [VL1,VL2,VL4]
          vnf: [VNF1,VNF2,VNF3]
        targets: [Forwarding path1, Forwarding path2]

      VNFFG2:
        type: tosca.groups.nfv.vnffg
        description: forwarding graph 2
        properties:
          vendor:
          version:
          vl: [VL1,VL3,VL4]
          vnf: [VNF1,VNF2]
        targets: [Forwarding path3]
```

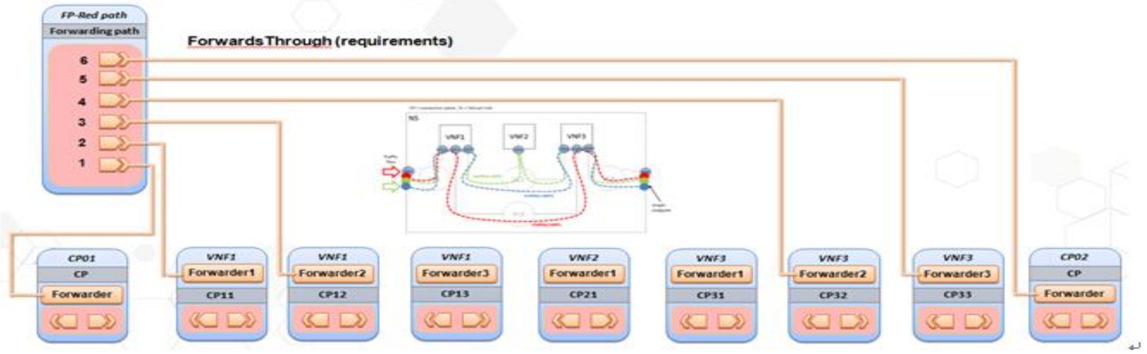
CP = connection point, VL = Virtual Link

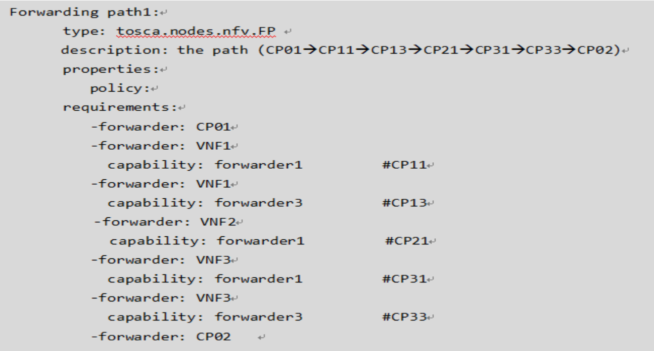


Source: xxx

NFP

Network forwarding path as defined by **ETSI NFV** is an order list of connection points forming a chain of network functions (VNFs or PNFs). A new "Forwarder" requirement is defined in this specification to model the network forwarding path by using ordered list of multiple "Forwarder" requirements. Each "Forwarder" requirement points to a single connection point.



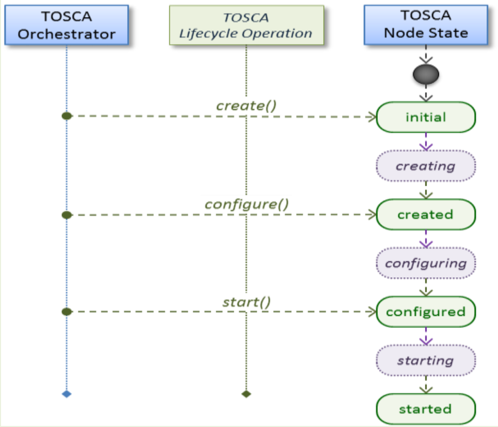


```
Forwarding path1:
      type: tosca.nodes.nfv.FP
      description: the path (CP01→CP11→CP13→CP21→CP31→CP33→CP02)
      properties:
         policy:
      requirements:
         -forwarder: CP01
         -forwarder: VNF1
           capability: forwarder1          #CP11
         -forwarder: VNF1
           capability: forwarder3          #CP13
          -forwarder: VNF2
           capability: forwarder1          #CP21
         -forwarder: VNF3
           capability: forwarder1          #CP31
         -forwarder: VNF3
           capability: forwarder3          #CP33
         -forwarder: CP02
```

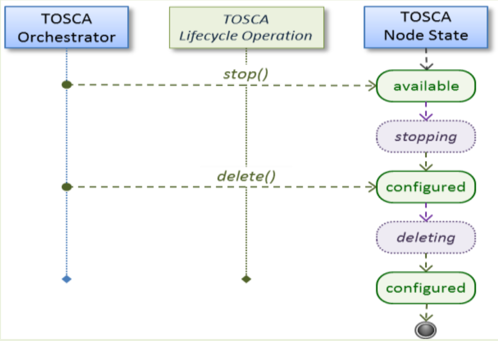
TOSCA Orchestrators - Standardized Lifecycle Sequencing

Lifecycle.Standard

Deploy Sequencing

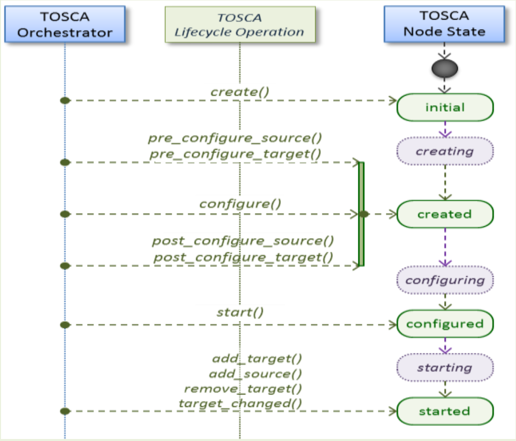


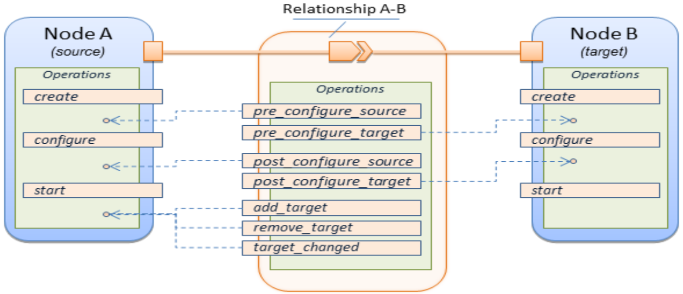
Undeploy Sequencing



Source-Target Sequencing

Lifecycle.Configure





Combined Sequencing

TOSCA Resources – Learn More

- TOSCA Technical Committee Public Page (latest documents, updates, and more)
  - https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca
- OASIS Channel (all standards) **or** TOSCA YouTube Playlist
  - https://www.youtube.com/user/OASISopen **or** http://bit.ly/1BQGGHm
- TOSCA Simple Profile in YAML v1.0 (latest committee approved draft)
  - http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.0/TOSCA-Simple-Profile-YAML-v1.0.pdf
- TOSCA Simple Profile for NFV v1.0 (latest committee approved draft)
  - http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/csd02/tosca-nfv-v1.0-csd02.pdf
- Contact the Technical Committee Co-Chairs:
  - **Paul Lipton, paul.lipton@ca.com**
  - **Simon Moser, smoser@de.ibm.com**
- Today's Presenters from the TOSCA TC:
  - **Shitao Li, lishitao@huawei.com**
  - **Matt Rutkowski, mrutkows@us.ibm.com**
  - **Chris Lauwers, lauwers@ubicity.com**
  - **Sridhar Ramaswamy , sramasw@Brocade.com**
  - **Sivan Barzily, sivan@gigaspaces.com**

TOSCA

An Open Standard for Business Application Agility and Portability in the Cloud

Q&A

Start Blueprinting Your Cloud Apps in TOSCA now!

25